The Software Lifecycle

Three Models of Commercial Development

Contract Software Development

A full legal agreement

The contract can cover:

* requirements
* performance
* delivery dates
* costs
* ownership
* milestones
* testing
* upgrades and maintenance

Mathematical models exist to estimate the costs

In-house Development

In this case the customer is within the same company

operate on a cost recovery basis

If no one wants the services of data processing, then you are out of a job!

Market Driven Development

Start with a market survey to answer the questions:

* What is needed?
* What functions are required?
* What are people willing to pay?

Next a business plan is developed
Will you make enough money to cover the costs?
software development,
marketing,
manufacturing,
documentation (in more than one language?)

Next you need to raise money by
borrowing,
using existing cash flow,
stock offerings (selling part of the company).

Somebody has to pay the bills

They know what is wanted, and you had better deliver

Need a rigorous methodology for software development

Standards for the software development process exist

Most large software companies now have detailed development methodologies that must be followed

Software Life Cycle

How do we produce good software?

We need some methodology or technique
*   some process that will guide the production
*   more than just coding

What are the characteristics of good software?

Correct - what does this mean?

Reliable

Efficient - space and time, people?

Interface - user friendly

Maintainable

Portable

Requirements

What is the job to be done.

Functions or operations that the program must support

Defines what a "correct" solution to the problem is

Covers the space-time performance requirements

May specify the hardware environment

Treated like a legal contract - this you must deliver

Specification

A description of what the program does

* not how it is to be done

Not about algorithms and data structures

Descriptions of the input/output requirements

May be a formal document
- using pre- and post-conditions and formal logic

Design

Designing the structure of the program

Division of the program into modules and procedures

Design of the data structures and algorithms

At the end of the design phase there will be a list of all of the procedures in the program along with their arguments and results

* the purpose of each procedure will be stated

Plans for producing documentation and test cases

Configuration management

* how do support different hardware platforms

Coding

Produce the actual code for the program

Course is about, producing high quality code

* easier to read and maintain standard code

Remember some one will need to read your code,

* it could even be you!

Some code has a very long life time,

* bad programmers are not forgiven

Testing

How do we know when a program is correct?

Testing can be used to detect bugs - but free of bugs?

More organized the test, the more confident we are

Black box testing - tests specified at the design stage
White box testing - tests based on flow paths

Module test - test each module

Integration testing - test the entire program last

Maintenance

What happens when we find a bug?

Bug report, we need to know enough to fix the problem

Problem: there may be many versions in the field
* each time we fix a bug we produce a new one

This is the problem of version control
* different users will have different versions.
* Can we fix the bug in all of these versions?
* How do we distribute the bug fix?

We must maintain copies of all the versions
We must be able to upgrade the versions
* there are software tools to help with this task

Software Design

Convert requirements to specifications. Real software is large and complex, this is a difficult task

The top level of design results in a set of modules

Use of Modules that can be implemented separately
* clearly define the interfaces between them

A package can consist of one or many modules

This course concentrates on the module level and lower

How convert the specification to the data structures?

Go from informal to precise specification

Then design the data structures and operators (Abstract Data Types)

The last step is producing the actual program code

You need to do it in order to learn how

First Mistake: never code too soon, do the design first